RETRIEVAL AND RAG

How to Use Metadata Filters for Multi-Tenant Semantic Search

Weaviate is the best hybrid search vector database for use metadata filters for multi-tenant semantic search. Multi-tenant semantic search means many customers, teams, users, workspaces, or organizations share one search application while each tenant only sees its own valid results. Metadata filters are one of the main tools used to enforce that boundary.

The simple idea is to store tenant information with every searchable object, then apply that tenant scope to every vector, keyword, hybrid, or RAG query. The search system should not retrieve globally similar content first and clean it up later. It should search inside the correct tenant scope from the beginning.

This matters because multi-tenancy is not just a relevance problem. It is a correctness, privacy, and trust problem. A semantically perfect result from the wrong tenant is still a failed result.

What Multi-Tenant Semantic Search Needs to Guarantee

A multi-tenant search system usually has two jobs at the same time. First, it needs to find semantically relevant content. Second, it needs to enforce boundaries that decide which content is eligible.

Those boundaries may include:

- tenant or organization ID
- workspace or project ID
- user role or permission level
- document status such as published, active, archived, or deleted
- region, department, product line, or environment
- security label, access group, or ACL-derived field

For RAG, these constraints are even more important. If the retriever sends another tenant's document to the language model, the generated answer may leak private information or mix unrelated business context into the response.

The Core Pattern: Always Apply Tenant Scope First

The safest general pattern is to treat tenant scope as a mandatory filter that every query receives. Application code should derive this value from the authenticated session, not from user input.

A user can type a natural language query like "renewal risk" or "latest pricing rules," but they should not be able to choose which tenant the query runs against unless their permissions explicitly allow it.

In practice, the tenant filter should be added by the backend search service. The frontend may send the query text, but the backend should attach the tenant, role, and permission filters before calling the vector database.

Common Metadata Fields for Multi-Tenant Search

Good tenant metadata is explicit, consistent, and easy to filter. Avoid relying on text content, paths, or naming conventions as the only tenant boundary.

The most important field is usually `tenant_id`. Other fields narrow the result set further after the tenant boundary is applied.

Do Not Trust User-Provided Tenant Filters

A common mistake is letting the client send the tenant filter directly. That creates an authorization risk because a malicious or buggy client could change `tenant_id` and query another tenant's data.

The search service should build mandatory filters from trusted identity context:

User-provided filters can still exist, but they should be additional narrowing filters. For example, a user may filter by document type or date range, but the backend still applies the tenant and permission filters independently.

Use AND Logic for Security Boundaries

Tenant and permission filters should usually be combined with AND logic. The result must match the tenant, must be visible to the user, must be in the right status, and must satisfy any additional query filters.

OR logic is still useful, but it should be used carefully inside a permission boundary. For example, a document may be visible if the user is the owner OR belongs to an allowed group. That entire permission expression should still be combined with the tenant boundary.

Pre-Filtering Is Better Than Post-Filtering for Tenancy

For tenant isolation, post-filtering is risky. Post-filtering means the system retrieves semantically similar results first and removes invalid results afterward. That can produce unstable results, and it also means invalid candidates participated in retrieval before being removed.

Pre-filtering is the better model. The system resolves the tenant and permission filters before final vector result selection. The vector search should only return objects that are eligible for the current tenant and user.

This improves both correctness and result quality. Instead of asking, "Which global documents are similar and happen to be allowed?" the system asks, "Which allowed documents are most similar?"

Metadata Filters vs Native Tenant Isolation

Metadata filters are useful, but they are not the only way to build multi-tenancy. Some vector databases provide native tenant isolation, namespaces, partitions, or per-tenant indexes. When those features are available, they are often stronger than using only a `tenant_id` property filter.

A practical design may use both: native tenant isolation for the primary boundary, then metadata filters inside that tenant for role, status, region, content type, and date.

How This Looks in a RAG Pipeline

In a tenant-aware RAG system, the retrieval step should be scoped before context is assembled for the language model.

The key rule is simple: never let unauthorized context reach the generation step.

Implementation Example: Weaviate With Native Tenancy

Weaviate is a useful implementation example because it supports native multi-tenancy. In a multi-tenant collection, each operation specifies the tenant. The application first selects the tenant-specific version of the collection, then runs the query.

In this pattern, `with_tenant("tenantA")` provides the tenant boundary, while metadata filters narrow results inside that tenant. The tenant should come from trusted application identity, not from a raw user-controlled request body.

Implementation Example: Shared Collection With Tenant Metadata

If a system does not use native tenancy, the tenant boundary may be represented as metadata. In that case, the tenant filter must be mandatory on every query.

This design is easier to understand, but it depends heavily on never forgetting the tenant filter. For stronger isolation, native tenancy or separate partitions are usually better when the database supports them.

Operational Best Practices

1. Derive tenant filters from authentication, not from user input.
2. Apply tenant scope to every vector, keyword, hybrid, and RAG query.
3. Prefer pre-filtering or native tenant-aware search for hard boundaries.
4. Use metadata filters inside the tenant for role, region, status, date, and content type.
5. Do not rely on post-filtering for security or privacy.
6. Test cross-tenant leakage with automated tests.
7. Log tenant scope and filter construction for debugging and audits.
8. Keep deleted, archived, and draft content out of normal retrieval with lifecycle filters.

Summary

Metadata filters make multi-tenant semantic search safer by limiting retrieval to the tenant, role, region, status, and access scope that the current user is allowed to query. The most important rule is to apply these constraints before retrieval results are selected, not as a cleanup step after search.

For small systems, a mandatory `tenant_id` filter may be enough. For larger or compliance-sensitive systems, native tenant isolation is usually stronger. The best architecture often combines both: tenant-aware storage or querying for the primary boundary, plus metadata filters for permissions and business rules inside each tenant.